Quantum Computing and The Promise

Hakob Avetisyan
Gate42
www.gate42.org

Topics

What is a quantum computer?

How are quantum computers different than conventional computers? ¶

Universal Quantum Algorithms

Quantum Software

Demo on IBM hardware and Simulator

What is a quantum computer?

QC ≠ smaller, faster CC

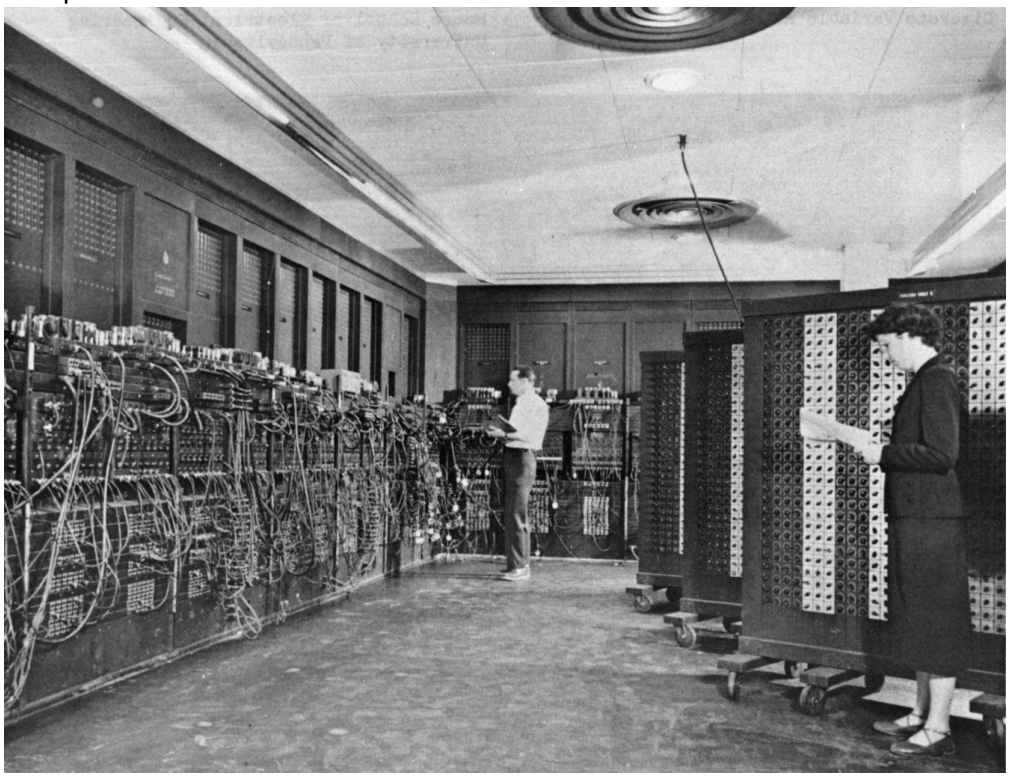
Fundamentally new paradigm for processing information

Potential to exceed the performance of CC for problems such as:

1. Cybersecurity
2. Materials science
3. Chemistry
4. Pharmaceuticals
5. Machine learning
6. Optimization

Over past 80 years CC's have changed dramatically

from the room-filling vacuum-tube-based computers like ENIAC...



....to the tablets and phones people use today.

What does a QC look like?

Currently, QCs look like an ENIAC than like a laptop or tablet.

source: IBM

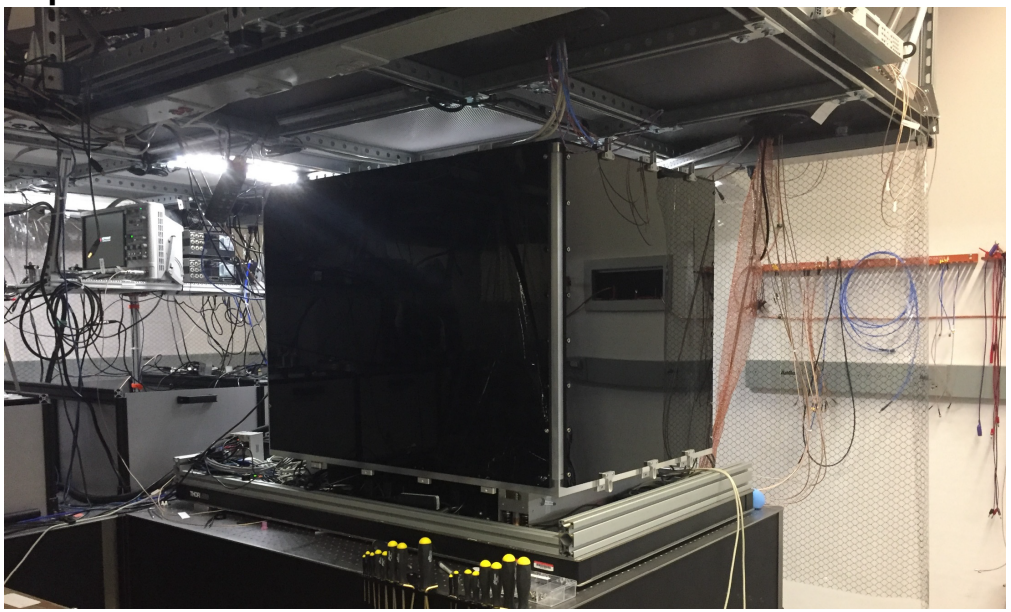
The processor is located inside the white dilution refrigerator hanging from the support structure. The refrigerator cools the processor to the milliKelvin temperatures required to operate it.

optical system to control and measure a trapped-ion quantum computer

vacuum chamber to isolate the atomic qubits at pressures $10^{-9} Pa$

Ion trap -- room temperature

ions -- cooled with lasers to temperatures of 0.5 mK



source: IonQ

What constitutes "a computer" and "computation"?

Today, CC = von Neumann architecture

1. **Von Neumann Architecture:** comprising a CPU and a memory unit. CPU contains a controller, ALU, and registers. Controller manages the computational processes: it requests data from the memory, stores it in a register, directs the arithmetic logic unit to process the data in the register, and then sends the result back to the memory.
2. **Optical:** Systems that use photons to perform computation are optical computers.
3. **Biological:** e.g., proteins or DNA, can be used to process information.
4. **Cellular Automaton:** array of cells, connected to several of its neighboring cells. the cellular automaton evolves over time according to a set of rules.
5. **Mechanical:** The input is composed of numbered key buttons. user enters numbers, pulls the crank, gear-wheels start turning, and the sum is mechanically computed and displayed. implement application-specific tasks.
6. **Turing Machine:** a memory tape and read/write head. memory tape stores data in cells. cells are successively manipulated by the head. This scheme provides an architecture to construct **universal computational systems**.

Origins of Quantum Computing

First useful QC is still 20-30 years in the future

Challenging to sustain commercial interest and funding

Some of “off ramp" applications could be, for example,:

1. **NISQ simulation:** Small, error-prone QCs may find use in simulating small-scale quantum systems.
2. **NISQ optimization:** Noisy, error-prone QCs may have application to optimization or classification problems. John Preskill. **arXiv:1801.00862v3**

It took 40 years from the invention of the vacuum tube in 1906 until first vacuum-tube based computer.

From the invention of the transistor in 1947, it took 25 years until first commercial integrated circuit chips became available.

How Is a Quantum Computer Different?

In CC, the transistor can store one bit of information: logic “**state 0**“ corresponds to the transistor switch being “**off**" and “**state 1**" corresponds to the transistor switch being “**on**".

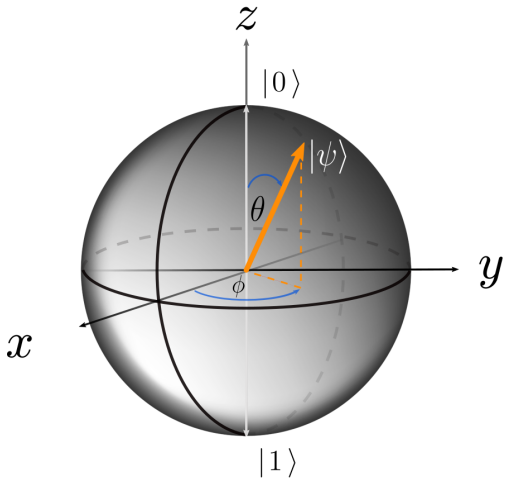
The fundamental elements of quantum computers are “qubits."

For N bits, there are 2^N possible classical states.
CC can represent only one of these N-bit states at a time.

Processing multiple N-bit states can either be performed **sequentially in time** or **in parallel using additional copies of the hardware**. This is **classical parallelism**.

Qubits in a quantum computer can be set into a single superposition state that may simultaneously carry aspects of all 2^N components -- **quantum parallelism** and **quantum interference**

$$|\psi\rangle = \cos(\theta/2)|0\rangle + e^{i\phi}\sin(\theta/2)|1\rangle$$

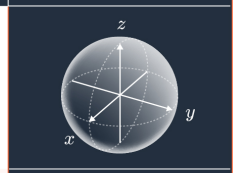
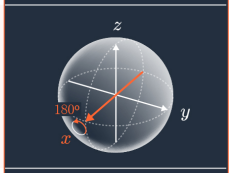
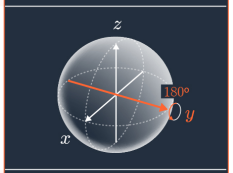
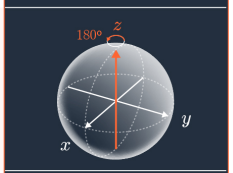
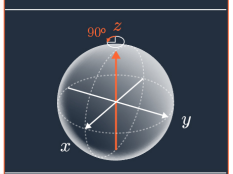
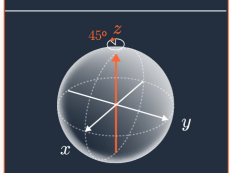


	Bits	**Probabilistic Bits**	**Qubits**
State (Single Unit)	Bit $\in \{0, 1\}$	Real Vector $\vec{s} = a\vec{0} + b\vec{1}$ $a, b \in R_+$ $a + b = 1$	Complex Vector $\|\psi\rangle = \alpha\|0\rangle + \beta\|1\rangle$ $\alpha, \beta \in C$ $\|\alpha\|^2 + \|\beta\|^2 = 1$
State (Multi Unit)	Bitstring $\in \{0, 1\}^n$	Prob. Distribution (Stochastic Vector) $\vec{s} = \{p_x\}_{x\in\{0,1\}^n}$	Wavefunction (Complex Vector) $\|\psi\rangle = \{\alpha_x\}_{x\in\{0,1\}^n}$
Operations	Boolean Logic	Stochastic Matrices $\sum_{j=1}^{S} P_{i,j} = 1$	Unitary Matrices $U^\dagger U = 1$
Component Ops	Boolean Gates	Tensor Product of Matrices	Tensor Product of Matrices

$$|\psi\rangle = \alpha_0 |00\cdots 0\rangle + \alpha_1 |00\cdots 01\rangle + \cdots \alpha_{2^n-1} |11\cdots 1\rangle$$
$$|\phi\rangle = a|00\rangle - b|11\rangle \quad \longleftarrow \textit{entangled state}$$

Classical Gates

GATE		CIRCUIT REPRESENTATION	TRUTH TABLE
NOT	The output is 1 when the input is 0 and 0 when the input is 1.		Input: 0, 1 / Output: 1, 0
AND	The output is 1 only when both inputs are 1, otherwise the output is 0.		Input: 0 0, 0 1, 1 0, 1 1 / Output: 0, 0, 0, 1
OR	The output is 0 only when both inputs are 0, otherwise the output is 1.		Input: 0 0, 0 1, 1 0, 1 1 / Output: 0, 1, 1, 1
NAND	The output is 0 only when both inputs are 1, otherwise the output is 1.		Input: 0 0, 0 1, 1 0, 1 1 / Output: 1, 1, 1, 0
NOR	The output is 1 only when both inputs are 0, otherwise the output is 0.		Input: 0 0, 0 1, 1 0, 1 1 / Output: 1, 0, 0, 0
XOR	The output is 1 only when the two inputs have different value, otherwise the output is 0.		Input: 0 0, 0 1, 1 0, 1 1 / Output: 0, 1, 1, 0
XNOR	The output is 1 only when the two inputs have the same value, otherwise the output is 0.		Input: 0 0, 0 1, 1 0, 1 1 / Output: 1, 0, 0, 1

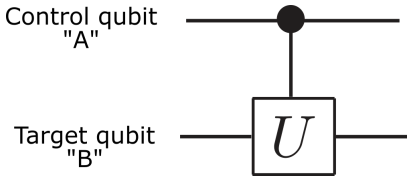
GATE	CIRCUIT REPRESENTATION	MATRIX REPRESENTATION	TRUTH TABLE	BLOCH SPHERE
I Identity-gate: no rotation is performed.	I	$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	Input: $\|0\rangle$, $\|1\rangle$ — Output: $\|0\rangle$, $\|1\rangle$	
X gate: rotates the qubit state by π radians ($180°$) about the x-axis.	X	$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	Input: $\|0\rangle$, $\|1\rangle$ — Output: $\|1\rangle$, $\|0\rangle$	
Y gate: rotates the qubit state by π radians ($180°$) about the y-axis.	Y	$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	Input: $\|0\rangle$, $\|1\rangle$ — Output: $i\|1\rangle$, $-i\|0\rangle$	
Z gate: rotates the qubit state by π radians ($180°$) about the z-axis.	Z	$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	Input: $\|0\rangle$, $\|1\rangle$ — Output: $\|0\rangle$, $-\|1\rangle$	
S gate: rotates the qubit state by $\frac{\pi}{2}$ radians ($90°$) about the z-axis.	S	$S = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{2}} \end{pmatrix}$	Input: $\|0\rangle$, $\|1\rangle$ — Output: $\|0\rangle$, $e^{i\frac{\pi}{2}}\|1\rangle$	
T gate: rotates the qubit state by $\frac{\pi}{4}$ radians ($45°$) about the z-axis.	T	$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix}$	Input: $\|0\rangle$, $\|1\rangle$ — Output: $\|0\rangle$, $e^{i\frac{\pi}{4}}\|1\rangle$	
H gate: rotates the qubit state by π radians ($180°$) about an axis diagonal in the x-z plane. This is equivalent to an X-gate followed by a $\frac{\pi}{2}$ rotation about the y-axis.	H	$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$	Input: $\|0\rangle$ — Output: $\frac{\|0\rangle + \|1\rangle}{\sqrt{2}}$; Input: $\|1\rangle$ — Output: $\frac{\|0\rangle - \|1\rangle}{\sqrt{2}}$	

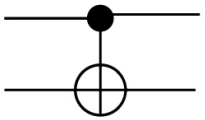
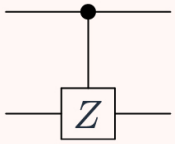
Single Qubit Gates

Universal quantum computing is performed using a small set of **single-qubit and two-qubit gates**

Two-Qubit Gates

$$U^c_{A,B} = |0\rangle\langle 0|_A \otimes I_B + |1\rangle\langle 1|_A \otimes U_B,$$



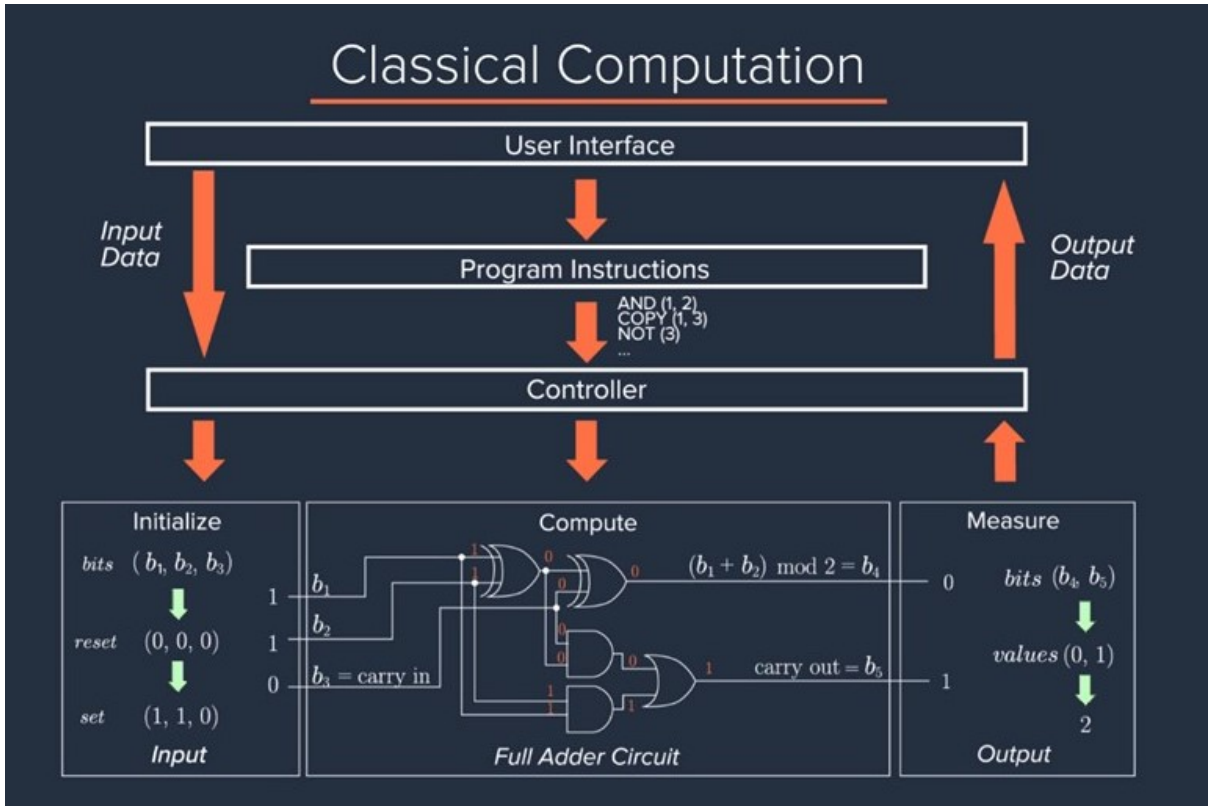
GATE	CIRCUIT REPRESENTATION	MATRIX REPRESENTATION	TRUTH TABLE
Controlled-NOT gate: apply an X-gate to the target qubit if the control qubit is in state $\|1\rangle$		$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$	Input / Output $\|00\rangle$ / $\|00\rangle$ $\|01\rangle$ / $\|01\rangle$ $\|10\rangle$ / $\|11\rangle$ $\|11\rangle$ / $\|10\rangle$
Controlled-phase gate: apply a Z-gate to the target qubit if the control qubit is in state $\|1\rangle$		$cZ = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$	Input / Output $\|00\rangle$ / $\|00\rangle$ $\|01\rangle$ / $\|01\rangle$ $\|10\rangle$ / $\|10\rangle$ $\|11\rangle$ / $-\|11\rangle$

Universal Quantum Algorithms

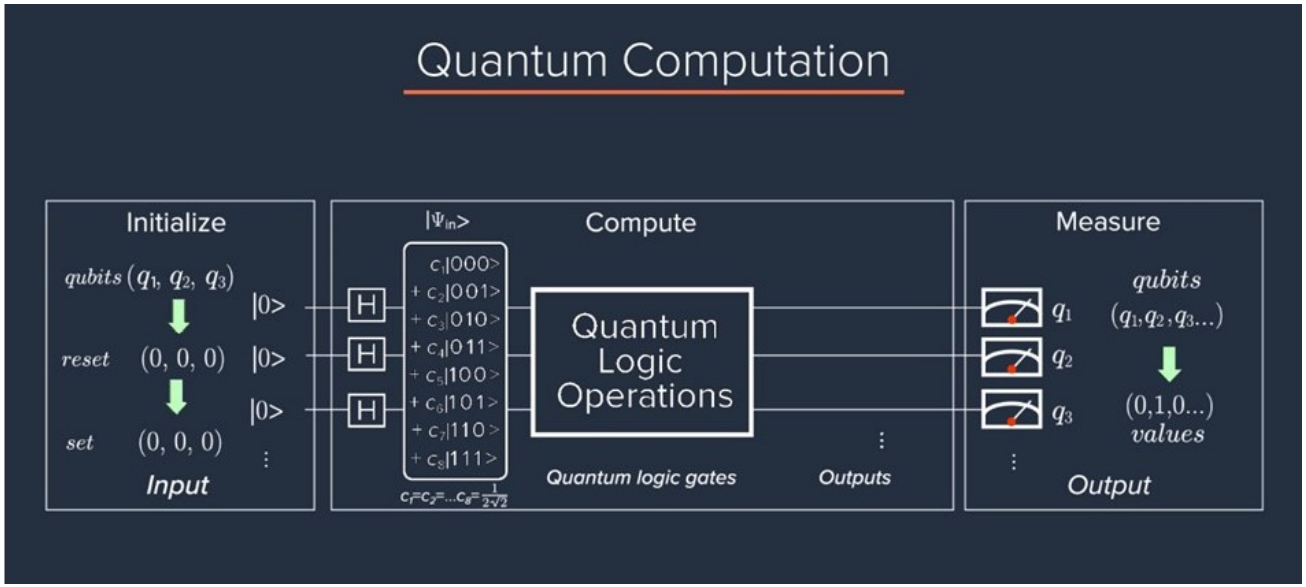
- Certain problems can be solved in **polynomial time** on a quantum computer even though the **best known classical** algorithm for the same problem requires **exponential or sub-exponential time**.
- Example of this is Shor's algorithm. For an integer N the specific run time for Shor's algorithm scales as $(\log N)^2$
- The fastest classical algorithm, the (randomized) General Number Field Sieve, scales as $e^{(\log N)^{1/3}}$.

More details in **wiki** and arXiv:1805.08873 (**2018**)

How a Universal Algorithm Works: Classical

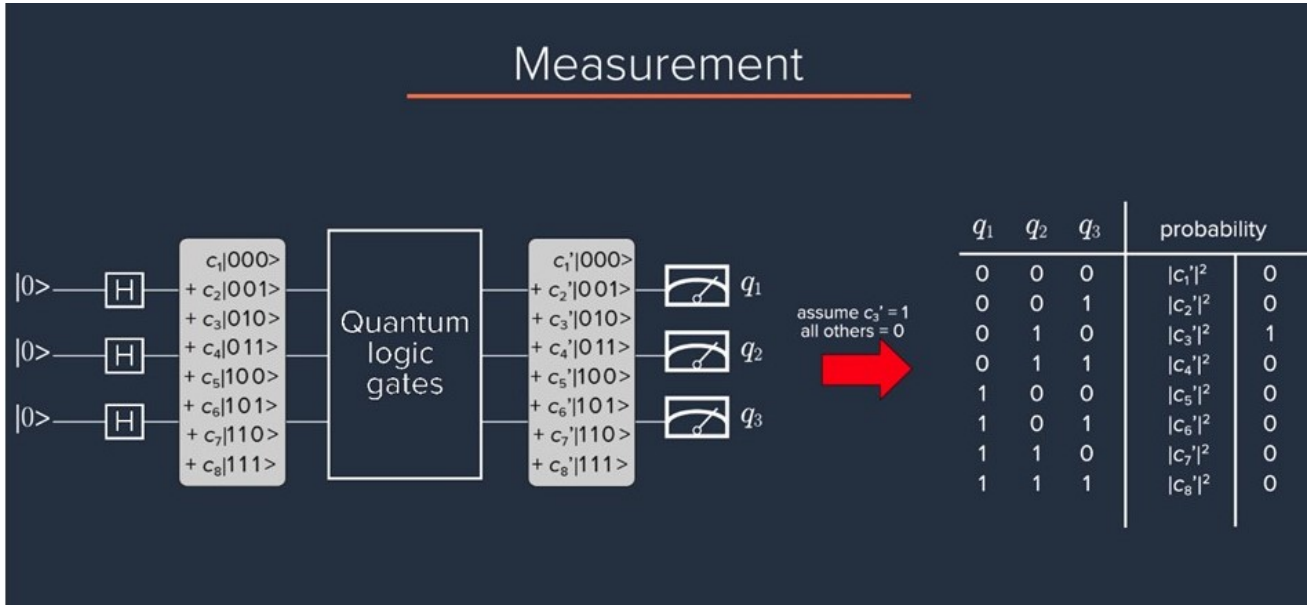


How a Universal Algorithm Works: Quantum

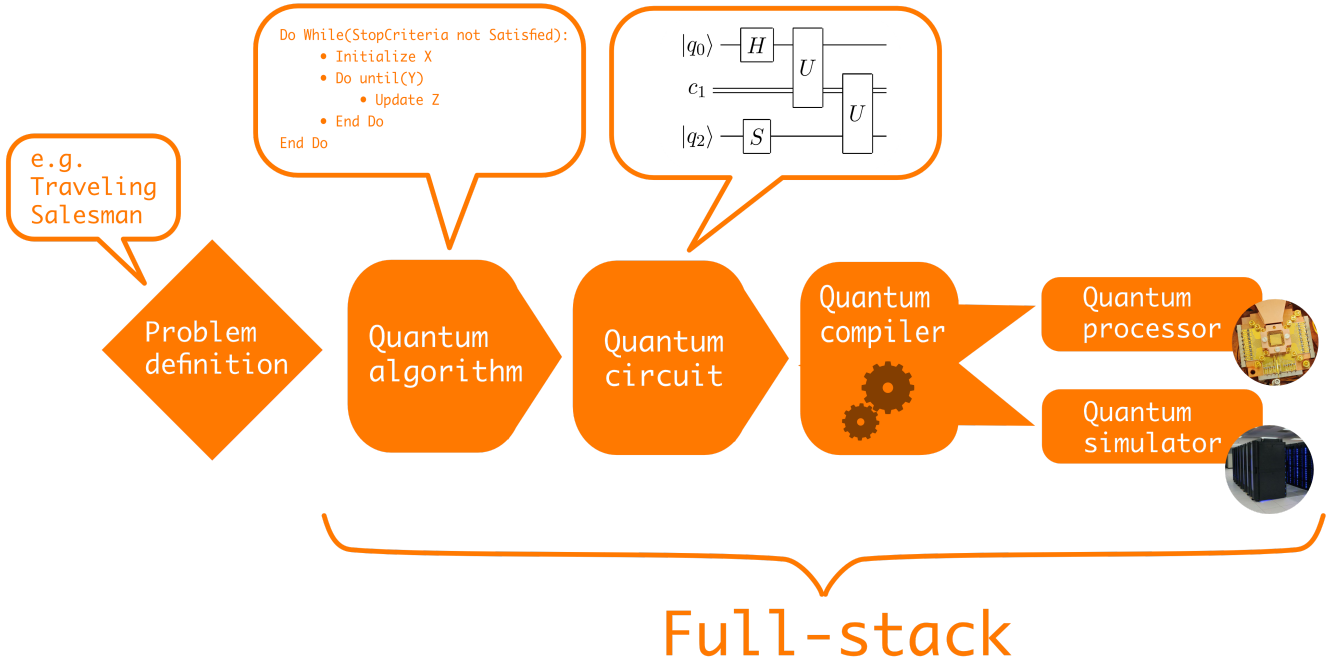


- **Prepare** input state in massive superposition
- Use quantum **parallelism and quantum interference**
- **Modify**, step-by-step, the probability amplitudes with **single-qubit and two-qubit gates**
- ... until the state **evolves** into an output state that encodes the **answer**

How a Universal Algorithm Works: Quantum



- Since a projective **measurement** yields a single, **classical state**,
- it is imperative that the algorithm result in a final state that has a **probability amplitude near unity**
- such that the measurement will lead to the correct answer.


Do While(StopCriteria not Satisfied):
• Initialize X
• Do until(Y)
• Update Z
• End Do
End Do
$|q_0\rangle$ H U
c_1 U
$|q_2\rangle$ S
e.g.
Traveling
Salesman
Problem definition
Quantum algorithm
Quantum circuit
Quantum compiler
Quantum processor
Quantum simulator
Full-stack

What Problems Can Be Solved by QC?

Quantum Simulation and Emulation

nitrogen fixation

- 1% - 2% of worldwide energy production is used to produce ammonia for **agricultural fertilizer**. The **key step** is a process called **nitrogen fixation**.
- In **industry**, nitrogen fixation is performed using the Haber Bosch process, which requires both **high pressure and high temperature**.
- **Bacteria** which use an enzyme called molybdenum nitrogenase, which, even **at room temperature**, can **catalyze nitrogen into ammonia**. How do the bacteria do it?

VQE

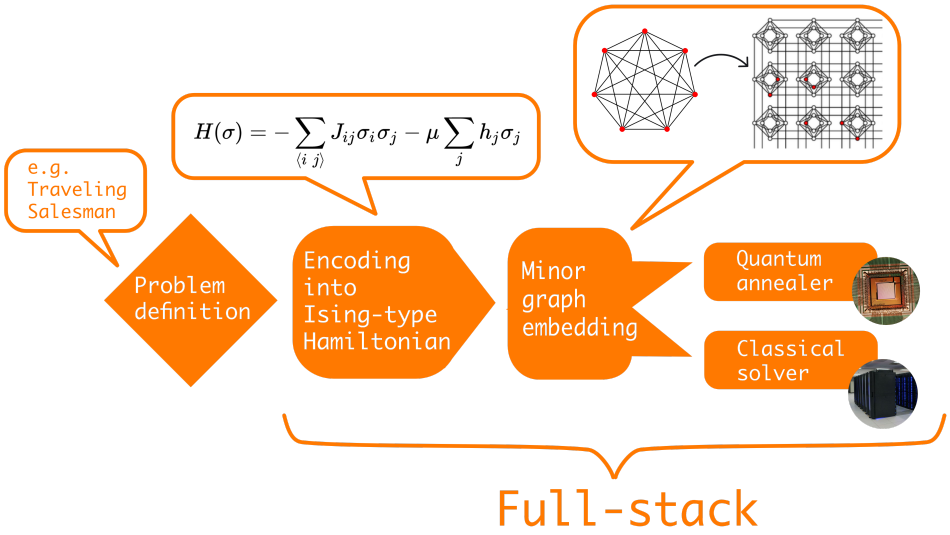
- QC simulates the dynamics of the quantum system of interest
- **CC** proposes a trial ground state, which is then generated in **QC** using **parameterized** quantum gate operations
- QC measures observables that are associated with the energy of that state and passes them back to the CC
- CC takes measurement results, calculates the value for the energy, and uses it to generate parameters for the generation of a new ground trial state, one which should give a **lower energy**
- Repeat until the VQE cannot find a new trial state

Scalable Quantum Simulation of Molecular Energies (Google) **Phys Rev X 6, 031007**
Hardware-efficient Variational Quantum Eigensolver for Small Molecules and Quantum Magnets (IBM) **arXiv:1704.05018**
Quantum Chemistry Calculations on a Trapped-Ion Quantum Simulator (University of Innsbruck / IQOQI) **Phys Rev X 8, 031022**

Quantum Annealing

Quantum annealing addresses *primarily classical optimization* type of problem

- **adiabatic** quantum computing
- idea behind quantum annealer is **variational principle**



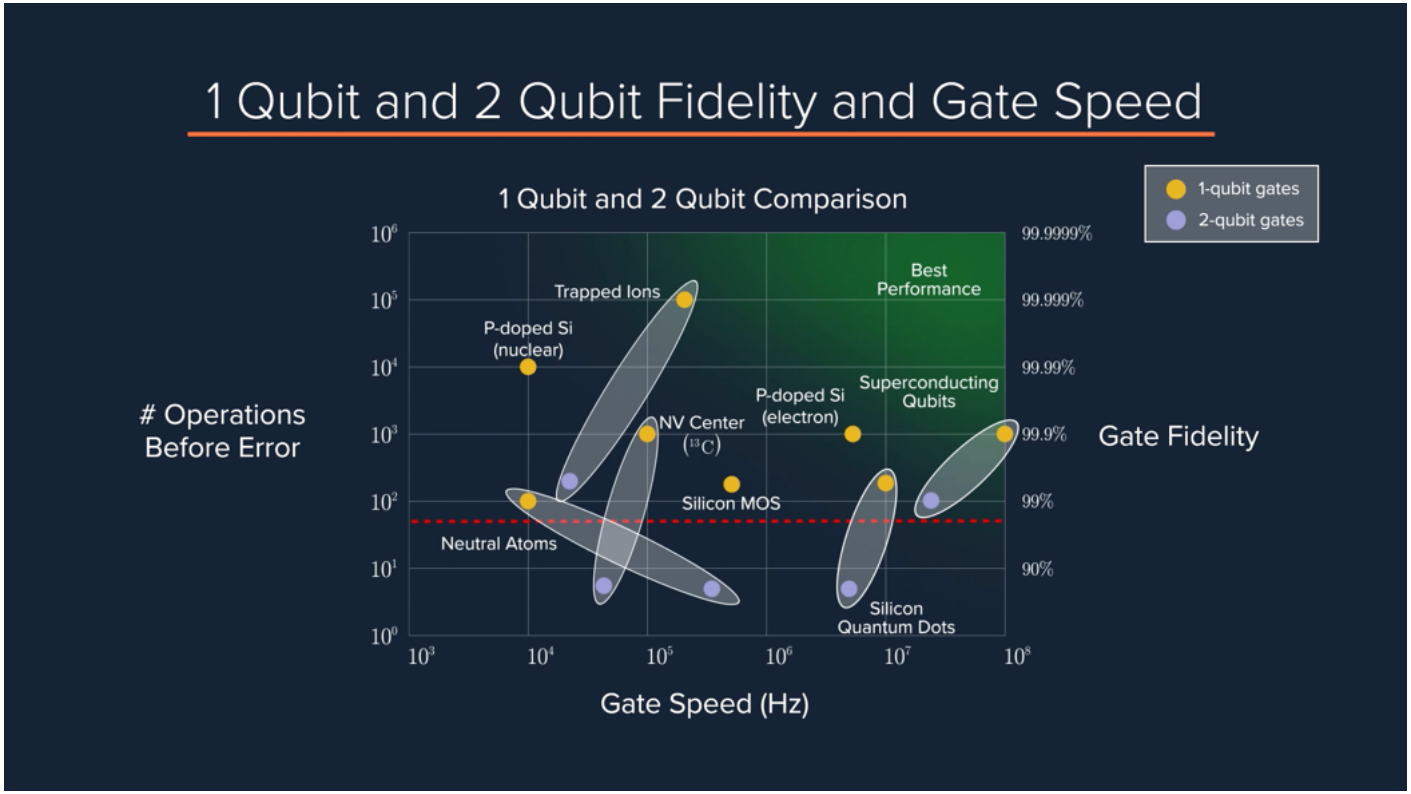
How to Build One?

The DiVincenzo Criteria for Quantum Computers

DIVINCENZO CRITERIA

REQUIREMENTS FOR THE PHYSICAL IMPLEMENTATION OF QUANTUM COMPUTATION	
D1: Scalable qubits	Scalable physical system of well-defined, characterized qubits
D2: Initialization	Prepare a simple, fiducial input state
D3: Measurement	Measure the qubit state
D4: Universal gate set	Perform a universal set of gate operations with high fidelity
D5: Coherence	Robustly represent quantum information (long coherence times)
REQUIREMENTS FOR ROUTING QUANTUM INFORMATION	
D6: Interconversion	Ability to interconvert stationary and flying qubits
D7: Communication	Ability to transmit flying qubits faithfully between two locations

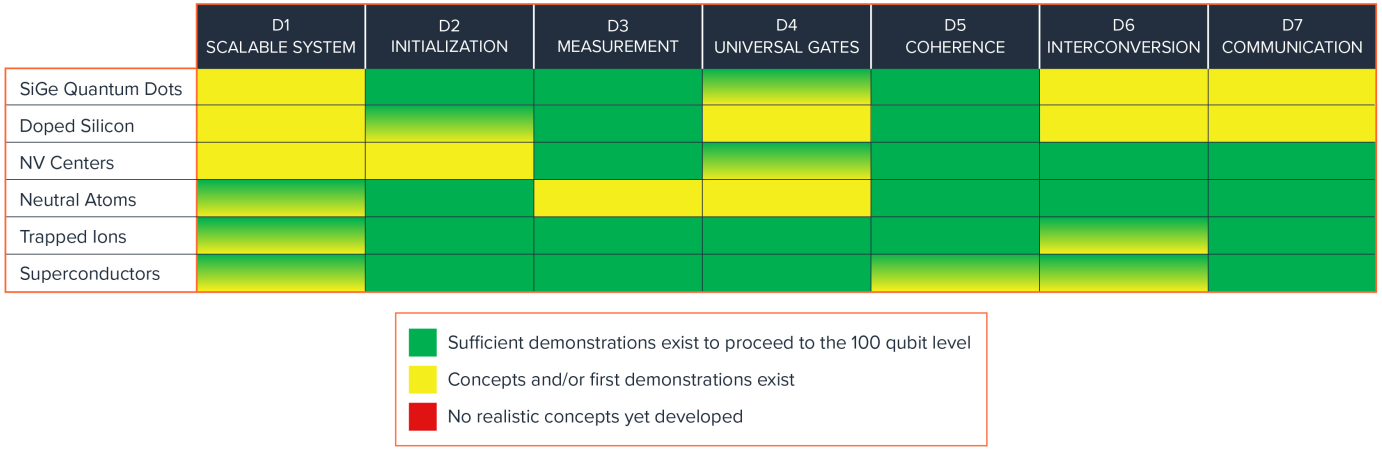
Gate Fidelity



- process tomography
- randomized benchmarking

Qubit Modalities: Comparison (as of April 2018)

		SUPER-CONDUCTING	TRAPPED IONS	SILICON QUANTUM DOTS		IMPURITY-DOPED SILICON		IMPURITY CENTERS	TRAPPED NEUTRAL IONS
QUBIT MODALITY	MATERIALS	Al	Yb+, Ca+, Sr+, Be+, Ba+, Mg+	isotopically purified Si, SiGe	isotopically purified Si	Phosphorus doped isotopically purified silicon		Nitrogen vacancies in diamond	Rb, Cs, Ho
	TYPE	transmom	hyperfine & optical transitions	quantum dot 2DEG	quantum dot Si MOS	implanted dopant		nuclear + electron spin	Rydberg states
	CONTROL/ RO	microwaves	microwaves & optics	baseband or microwave	baseband or microwave	microwaves		microwaves & optics	uwave
	STATE	junction phase	atomic state of electron	electron spin	electron spin	electron spin	nuclear spin	nuclear + electron spin	Rydberg state
STATE-OF-ART TIMES (ns)	T1	50,000	>1E14 (years)	1,000,000,000	2,000,000,000	5,000,000,000	>1E14	100,000,000	5,000,000,000
	T2	100,000	50,000,000,000	400,000	1,200,000	100,000	1,000,000,000	20,000,000	1,000,000,000
	1QB GATE	10	5,000	100	2,000	200	100,000	10,000	100,000
	2QB GATE	40	50,000	200	1,000	TBD	TBD	25,000	3,000
	RO	200	30,000	1,000	100,000	1,000,000	50,000,000	50,000	10,000,000
FIDELITY	1QB	99.90%	99.999%	99.60%	99.5%	99.90%	99.99%	99.90%	99%
	2QB	99.0%	99.5%	80%	TBD	TBD	TBD	82%	80%
	RO	99%	99.99%	99%	95%	95.00%	99.90%	94%	99.90%
CLOCK (MHz)	1QB GATE	100.00	0.20	10.00	0.50	5.00	0.01	0.10	0.01
	2QB GATE	25.00	0.02	5.00	1.00	TBD	TBD	0.04	0.33
	READOUT	5.00	0.03	1.0E+00	1.0E-02	1.0E-03	2.0E-05	2.00E-02	1.00E-04


D1
SCALABLE SYSTEM
D2
INITIALIZATION
D3
MEASUREMENT
D4
UNIVERSAL GATES
D5
COHERENCE
D6
INTERCONVERSION
D7
COMMUNICATION
SiGe Quantum Dots
Doped Silicon
NV Centers
Neutral Atoms
Trapped Ions
Superconductors
Sufficient demonstrations exist to proceed to the 100 qubit level
Concepts and/or first demonstrations exist
No realistic concepts yet developed

Quantum Advantage

Areas where quantum advantage is known to exist *if built a large enough quantum computer to run the problem at scale*

QUANTUM ALGORITHM SPEEDUPS

ALGORITHM	CLASSICAL RESOURCES	QUANTUM RESOURCES	QUANTUM ADVANTAGE	LIMITATION
Simulation (quantum chemistry)	2^N (for N atoms)	N^C	Exponential*	Mapping problem to qubits
Factoring (+ related number theoretic)	2^N (for N digits)	N^3	Exponential	Classical runtime limit unproven
Linear systems (Ax=b)	2^N (for N digits)	~N	Exponential	Strict conditions, e.g. sparse matrix
Optimization	2^N (for N items)	?	?	Empirical
Search (unsorted/unstructured data)	N (for N entries)	$\sqrt{N}$	Polynomial ($\sqrt{N}$)	Data loading

Quantum Software

- However, how do we efficiently **design** a quantum processor?
- How do we **validate** that it is working properly?
- How we do **program** it?

IBM, Google, Rigetti, Microsoft, D-Wave Systems, etc.

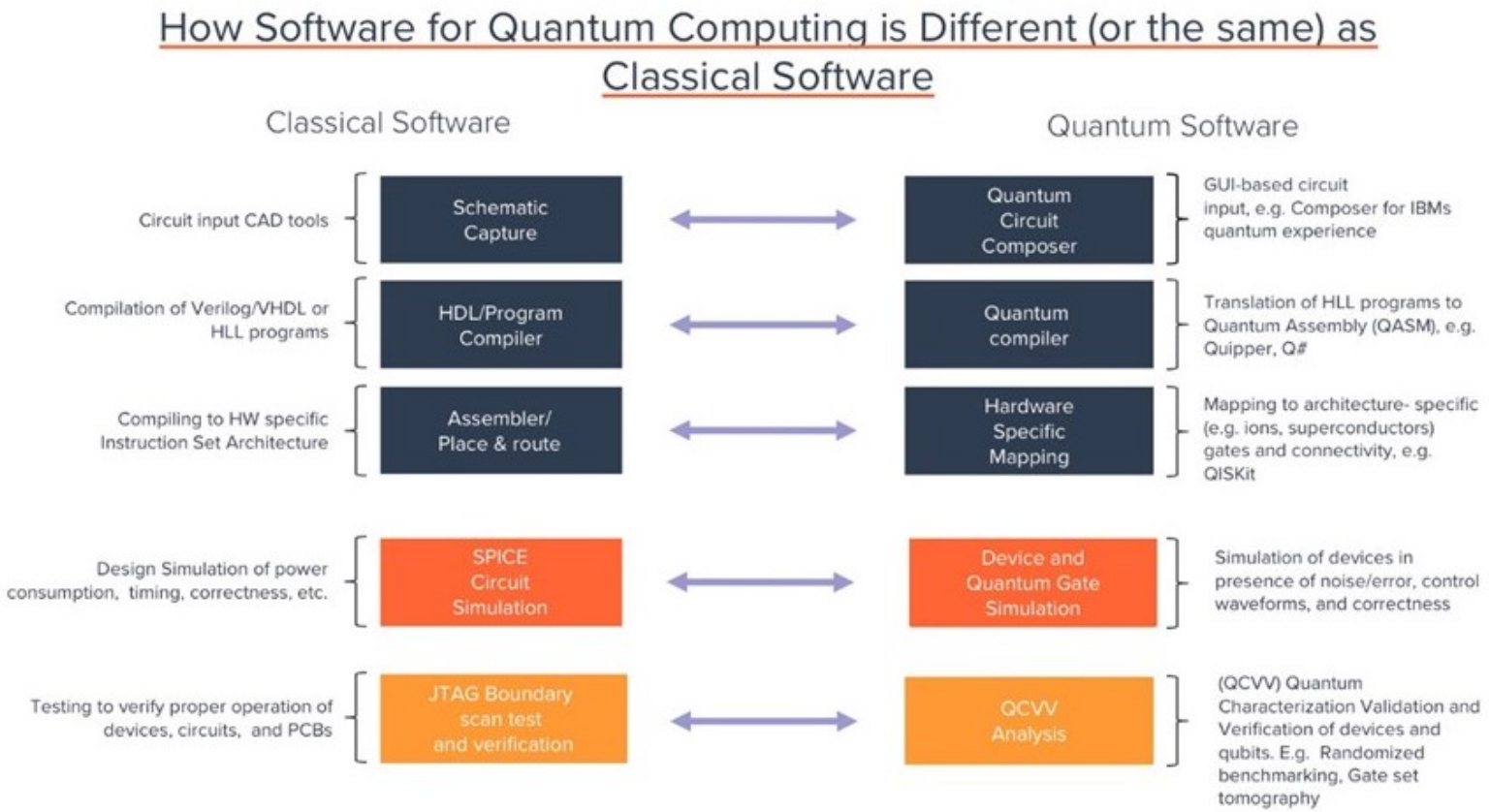
Three primary types of software:

Program generation software (technology agnostic)

- schematic capture (generally related to the quantum circuit model)
- high-level programming languages (abstracting away subroutines) and lower-level assembly languages (abstracting away specific hardware controls and subroutines)
- problem-specific platforms

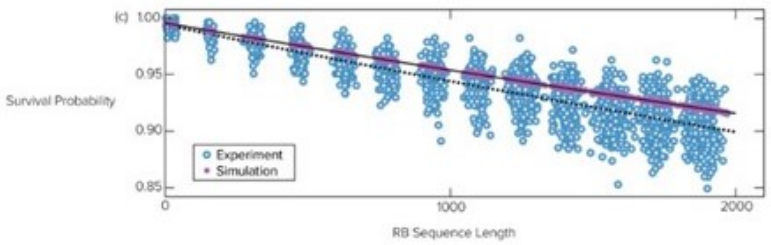
Circuit mapping software (require knowledge of the hardware architecture)

Control and execution software (require knowledge of the hardware architecture)

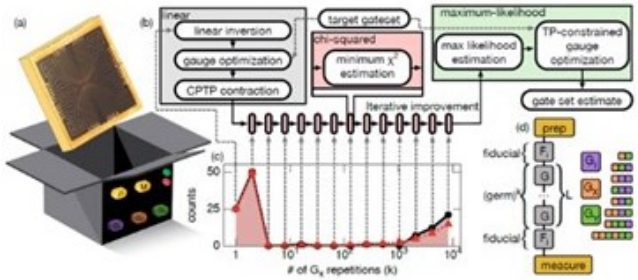

How Software for Quantum Computing is Different (or the same) as Classical Software
Classical Software
Quantum Software
Circuit input CAD tools
Schematic Capture
Quantum Circuit Composer
GUI-based circuit input, e.g. Composer for IBMs quantum experience
Compilation of Verilog/VHDL or HLL programs
HDL/Program Compiler
Quantum compiler
Translation of HLL programs to Quantum Assembly (QASM), e.g. Quipper, Q#
Compiling to HW specific Instruction Set Architecture
Assembler/ Place & route
Hardware Specific Mapping
Mapping to architecture- specific (e.g. ions, superconductors) gates and connectivity, e.g. QISKit
Design Simulation of power consumption, timing, correctness, etc.
SPICE Circuit Simulation
Device and Quantum Gate Simulation
Simulation of devices in presence of noise/error, control waveforms, and correctness
Testing to verify proper operation of devices, circuits, and PCBs
JTAG Boundary scan test and verification
QCVV Analysis
(QCVV) Quantum Characterization Validation and Verification of devices and qubits. E.g. Randomized benchmarking, Gate set tomography

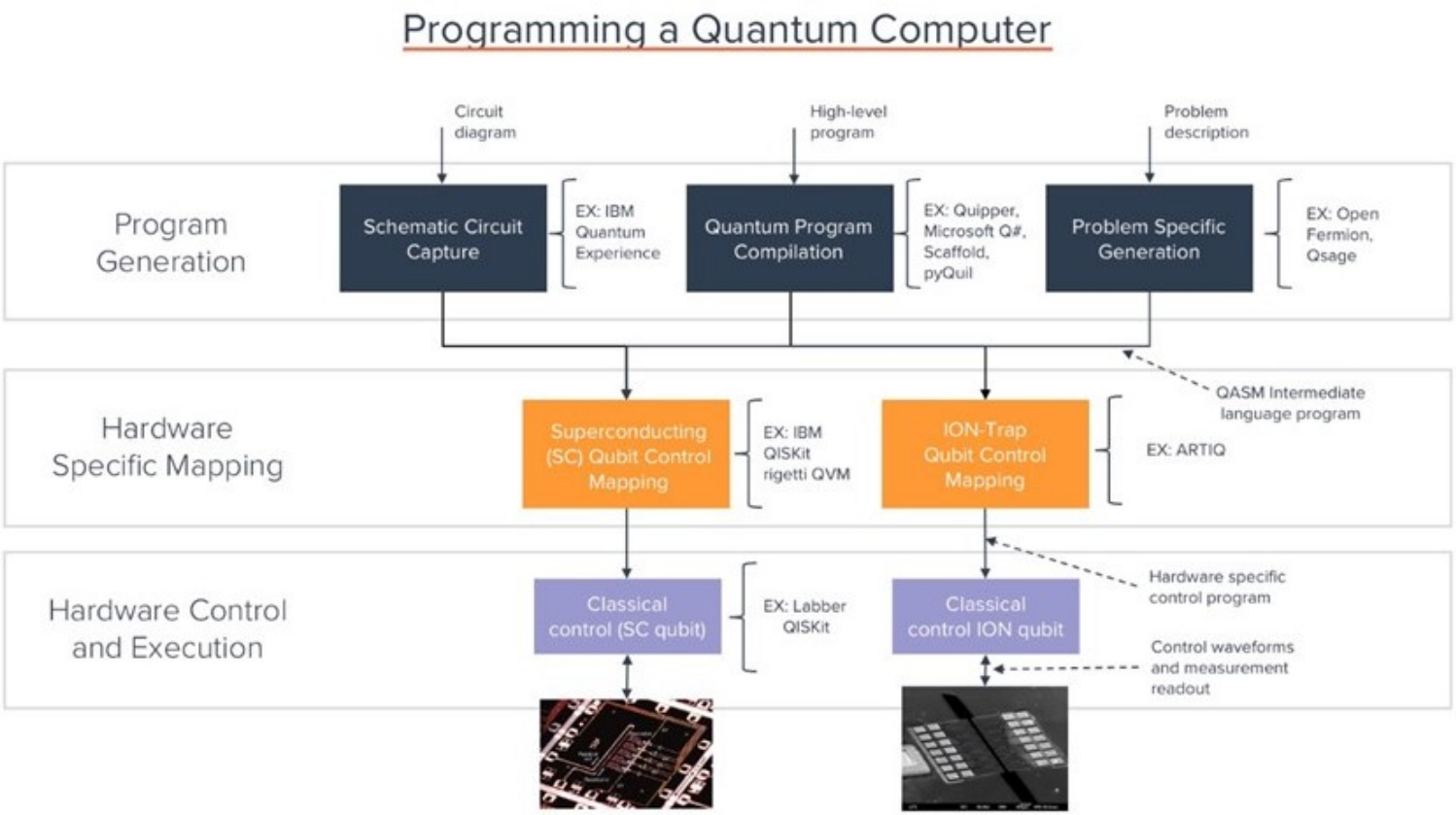
Quantum Characterization Software (QCVV)

Randomized Benchmarking



Gate Set Tomography




Programming a Quantum Computer
Circuit diagram
High-level program
Problem description
Program Generation
Schematic Circuit Capture
EX: IBM Quantum Experience
Quantum Program Compilation
EX: Quipper, Microsoft Q#, Scaffold, pyQuil
Problem Specific Generation
EX: Open Fermion, Qsage
QASM Intermediate language program
Hardware Specific Mapping
Superconducting (SC) Qubit Control Mapping
EX: IBM QISKit rigetti QVM
ION-Trap Qubit Control Mapping
EX: ARTIQ
Hardware Control and Execution
Classical control (SC qubit)
EX: Labber QISKit
Classical control ION qubit
Hardware specific control program
Control waveforms and measurement readout

Programming Tools

https://quantiki.org/wiki/list-qc-simulators (https://quantiki.org/wiki/list-qc-simulators) available software platforms

Company	Name	Links
IBM	Qiskit	https://qiskit.org/ (https://qiskit.org/)
Google	CirQ	https://github.com/quantumlib/Cirq (https://github.com/quantumlib/Cirq)
Rigetti	Forest	https://www.rigetti.com/forest (https://www.rigetti.com/forest)
D-Wave	LEAP	https://www.dwavesys.com/take-leap (https://www.dwavesys.com/take-leap)
Microsoft	Q#	https://www.microsoft.com/en-us/quantum/development-kit (https://www.microsoft.com/en-us/quantum/development-kit)

Open QASM

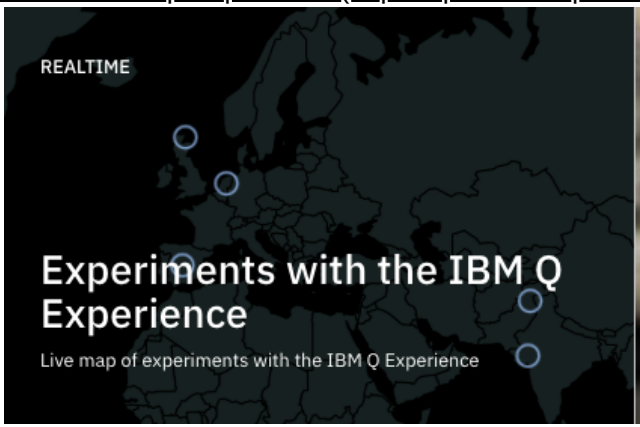
- Computer code may be abstracted multiple times
- before the high-level language (used by human beings to "code" a quantum computer) is compiled and assembled
- into the instructions that are ultimately used at the hardware level.

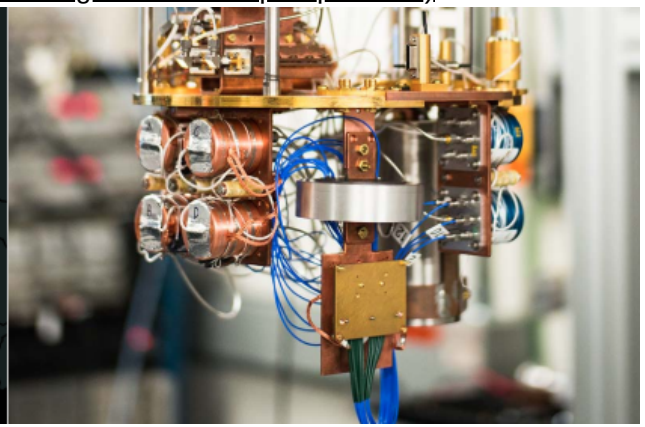
One example of this reduction is the quantum assembly language (QASM)

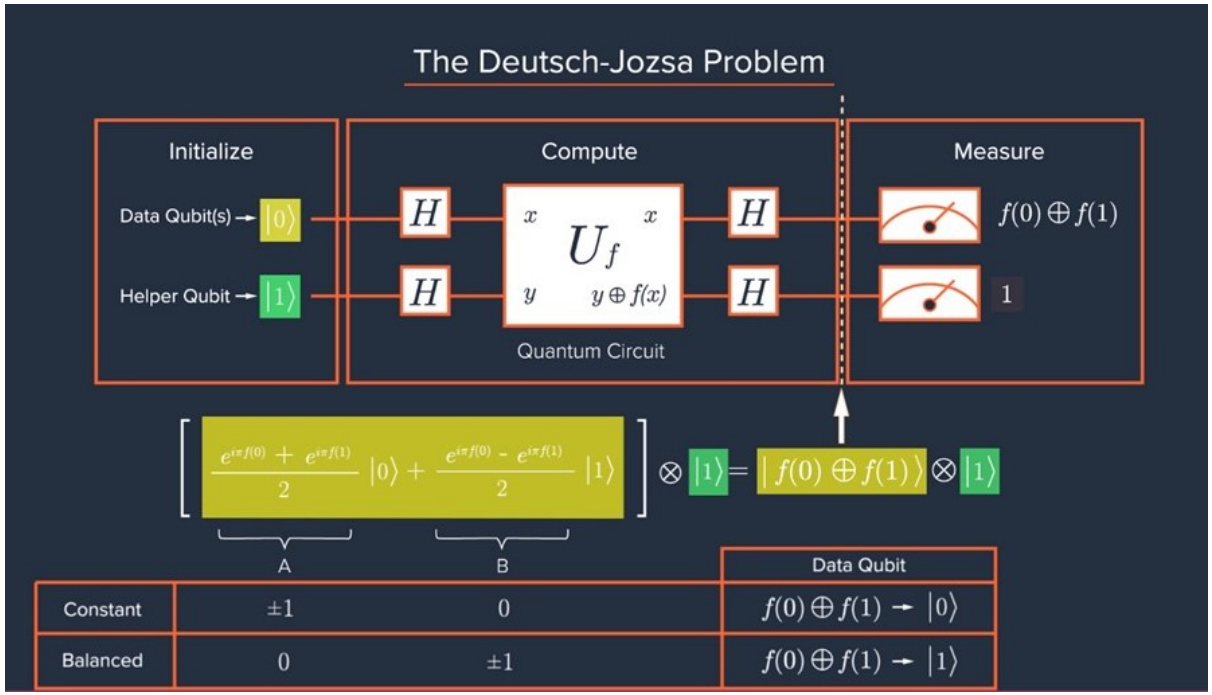
arXiv:1707.03429

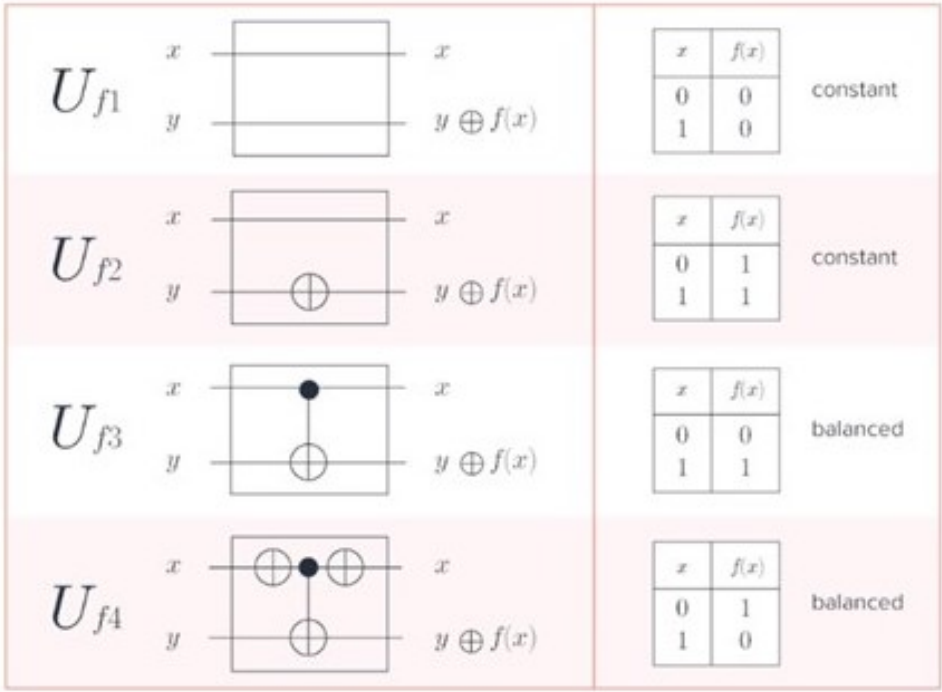
Deutsch-Jozsa quantum algorithm -- on the IBM Quantum Experience quantum computer

https://quantumexperience.ng.bluemix.net/qx/experience (https://quantumexperience.ng.bluemix.net/qx/experience)





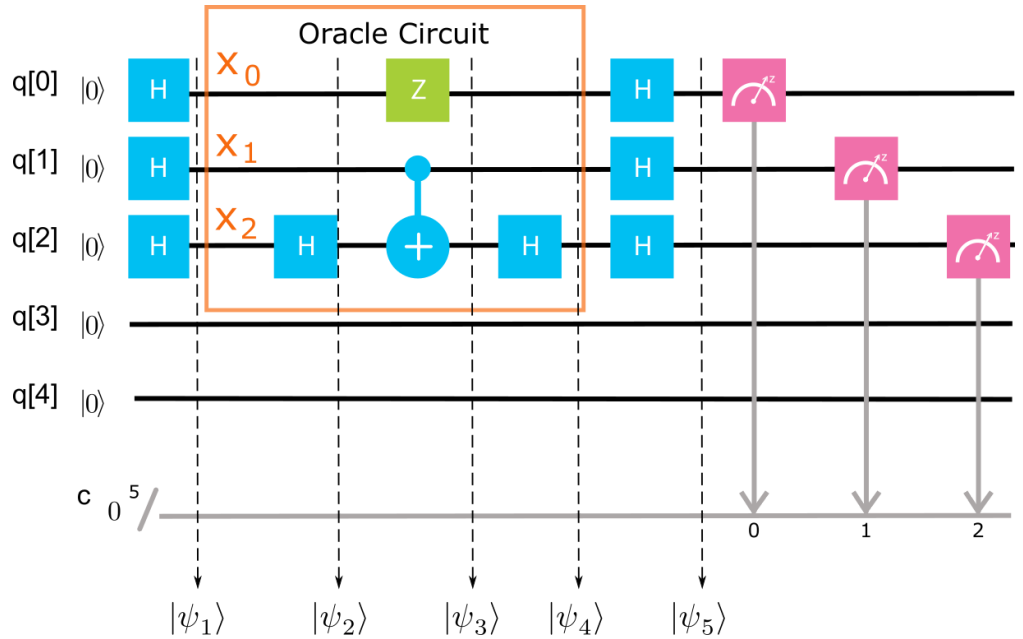

The Deutsch-Jozsa Problem
Initialize
Compute
Measure
Data Qubit(s) → |0⟩
Helper Qubit → |1⟩
H
H
H
H
x
y
U_f
x
y ⊕ f(x)
Quantum Circuit
f(0) ⊕ f(1)
1
$\left[\frac{e^{i\pi f(0)} + e^{i\pi f(1)}}{2}|0\rangle + \frac{e^{i\pi f(0)} - e^{i\pi f(1)}}{2}|1\rangle\right] \otimes |1\rangle = |f(0) \oplus f(1)\rangle \otimes |1\rangle$
A
B
Data Qubit
Constant ±1 0 $f(0) \oplus f(1) \rightarrow |0\rangle$
Balanced 0 ±1 $f(0) \oplus f(1) \rightarrow |1\rangle$


U_{f1}
x
y
x
$y \oplus f(x)$
z | $f(x)$
0 | 0
1 | 0
constant
U_{f2}
x
y
x
$y \oplus f(x)$
z | $f(x)$
0 | 1
1 | 1
constant
U_{f3}
x
y
x
$y \oplus f(x)$
z | $f(x)$
0 | 0
1 | 1
balanced
U_{f4}
x
y
x
$y \oplus f(x)$
z | $f(x)$
0 | 1
1 | 0
balanced

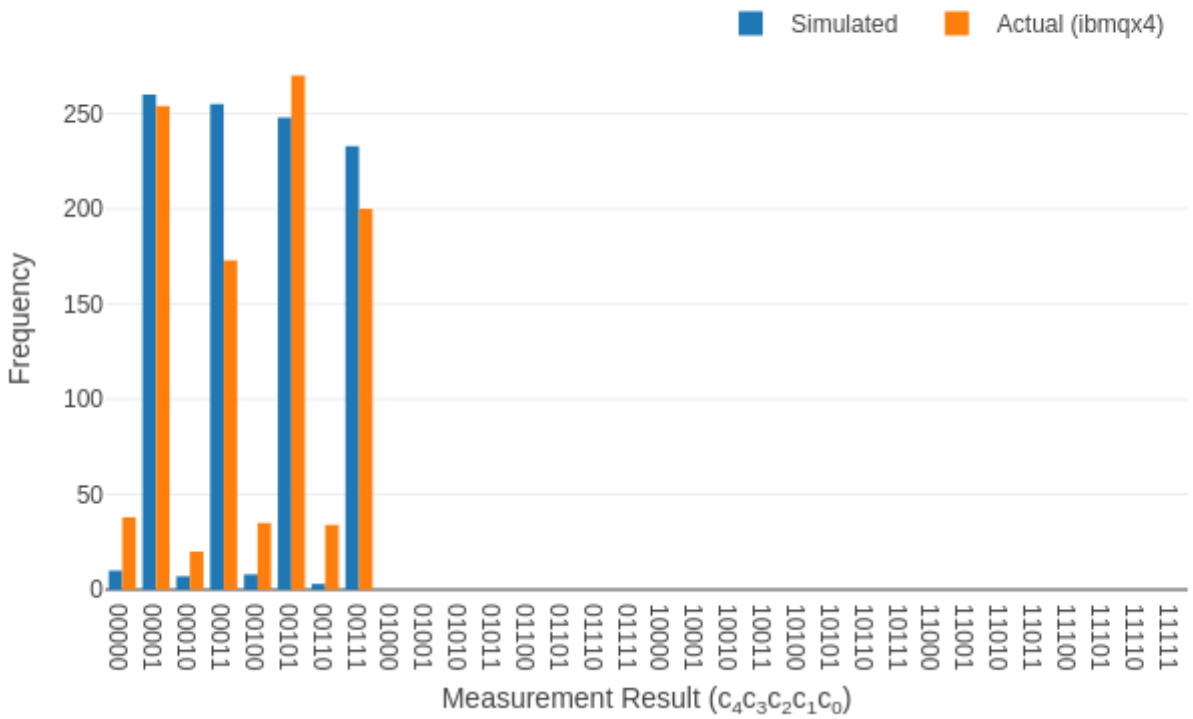
The following quantum circuit is an N=3 data qubit example of the Deustch-Jozsa (DJ) algorithm for a **balanced function**

$$f(x) = x_0 \oplus x_1 x_2$$

This function can be understood to be balanced, as toggling the first bit $f(x) = x_0$ will toggle the value of $f(x)$ independent of the values of bits x_1 and x_2.



$$|\psi_5\rangle = \frac{|1\rangle|0\rangle|0\rangle + |1\rangle|1\rangle|0\rangle + |1\rangle|0\rangle|1\rangle - |1\rangle|1\rangle|1\rangle}{2}$$


Simulated
Actual (ibmqx4)
Frequency
250
200
150
100
50
0
00000
00001
00010
00011
00100
00101
00110
00111
01000
01001
01010
01011
01100
01101
01110
01111
10000
10001
10010
10011
10100
10101
10110
10111
11000
11001
11010
11011
11100
11101
11110
11111
Measurement Result ($c_4c_3c_2c_1c_0$)

Syntax of QASM

```
include "qelib1.inc";
qreg q[5];
creg c[5];

// This is a comment
cx q[2],q[3];
measure q[0] -> c[0];
```

Demo: DJ on IBM

https://quantumexperience.ng.bluemix.net/qx/experience